CS246 Group Project: Chess

Group member: Ruozhen Gu(r8gu), Yining Song(ynsong), Zhengchun Wang(z383wang)

Plan Attack

Task	Decription	Assigned to	Estimated Completion Dates
Discuss the whole thoughts and ideas including UML and design pattern	Discuss about the superclasses and subclasses, the fields and methods they should have, what design pattern we need, the relationship between classes	Ruozhen Gu Yining Song Zhengchun Wang	March 23rd, Friday
Draw UML	Draw the UML diagram	Yining Song Zhengchun Wang	March 23rd, Friday
Discuss Implementation Questions	Discuss and analyse the implementation questions in the project file	Ruozhen Gu Yining Song Zhengchun Wang	March 23rd, Friday
Write down Plan Attack	Plan the tasks, the assigned people, and the deadline of each task	Yining Song Zhengchun Wang	March 23rd, Friday
Answer Implementation Questions	Write down the analysis and thoughts of each implementation question	Yining Song Zhengchun Wang	March 23rd, Friday
Write TextDisplay class files, GraphicDisplay class files, Subclasses files, Makefile	Write code and finish files	Yining Song	March 28th, Wednesday
Write Piece class files, Player class files	Write code and finish files	Zhengchun Wang	March 28th, Wednesday
Write Cell class files ,Board class files, main file	Write code and finish files	Ruozhen Gu	March 28th, Wednesday
Debug	Find and resolve problems and defects, and make sure no memory leak	Yining Song	March 30th, Friday
Test	Write some test cases to test the program	Ruozhen Gu	March 31st, Saturday

Check graphics output	Test the graphic output	Zhengchun Wang	March 31st, Saturday
Add bonus features	Discuss and try to implement some extra features	Yining Song Ruozhen Gu Zhengchun Wang	Apirl 1st, Sunday
Documentation	Complete documentation for code and design	Yining Song Ruozhen Gu Zhengchun Wang	Apirl 2nd, Monday
Final Design	Finish final design document and update UML	Yining Song Ruozhen Gu Zhengchun Wang	Apirl 2nd, Monday

Implementation Questions

Question 1. Chess programs usually come with a book of standard opening move sequences, which list accepted opening moves and responses to opponents' moves, for the first dozen or so moves of the game. Although you are not required to support this, discuss how you would implement a book of standard openings if required.

We will create an array of array, and each array in the array contains series of positions which stands for standard openings such as Giuoco Piano, King's Gambit, Sicilian Defense, etc.(https://www.dwheeler.com/chess-openings/) For example, Giuoco Piano starts with: 1. e4 e5 2. Nf3 Nc6 3. Bc4 Bc5. When one of the player is computer, the computer will go through this array, randomly pick a standard opening and stick with it. As for response to opponent's moves, every time when the computer want to move, it will first use the level 3 prefer avoids algorithm to avoid potential attack, then if there is no potential attack, it will continue move regarding with the standard opening it picked. Moreover, if we set restrictions on the first several moves, we will do relative modification. For example, if only rook can be moved in a specific moves, we will set that restriction in input, so that when user input a wrong piece, we will ask the user to input again. Also, if only a specific direction is allowed for that piece, we will check whether the coordinates the user input is the valid one.

Question 2. How would you implement a feature that would allow a player to undo his/her last move? What about an unlimited number of undos?

First let's try to add the functionality to undo the last step:
Inside Piece class, which is the mother/base class of all chess roles including queen, king … etc, it will have another 2 fields x_prev and y_prev which are used to record the last position of this piece. To be exact, when moving a chess, the current location (x and y) will be saved into x_prev and y_prev before they are updated with the new axis coordinator. For each of six roles, it will also have a field as a flag to indicate whether it's the last piece moved by this user. When a user hits undo feature, it will first call the corresponding player class (whether player1 or player2 hits the undo) and then check which chess has that flag on. When finding it, the current position (x and y) will be replaced with x_prev and y_prev while x_prev and y_prev can be set to -1 to indicate this chess is undoed. And the flag will be turned off in case the user wants to move another chess role.

Now let's try to solve the issue with unlimited undo features:
Obviously, adding x_prev and y_prev won't solve this issue. So we will need a stack to track all the movement of each chess. Exactly, there will be a vector of piece inside each player class. Every time if player1 moves a chess, we create a copy of that cell and push it into this vector before the current position x and y are changed.The same as player2. So imagine player1 now hits undo, the last element of this vector will be pop out from vector inside player1 class. We will use the function inside piece "get_name", "get_x" and "get_y" to see which role (king or queen or ...etc) exactly is moved and what its original coordinator is. Now we can go back to the corresponding player class and find that chess role and update its coordinator using the one we get above from that copied cell. The same for player2 and by using this vector to keep track of all movements of two players, we can support the feature of unlimited undo.

Question 3. Variations on chess abound. For example, four-handed chess is a variant that is played by four players (search for it!). Outline the changes that would be necessary to make your program into a four-handed chess game.

First of all, we will modify the class Cell and Board to enlarge the board size we have right now; the TextDisplay and GraphicDisplay will also have relative changes. Then we will create four player subjects when we run the game. We add two additional setup model in which one is 2 vs 2, and another is last standing win. Players can choose the mode of the game, and the play order of will be clockwise. If it is team-up model, then the six subclasses of Piece will be modified such that players cannot check their own team members. Also the Players class will have a pointer to their teammate to identify teammates and opponents. We will modify the main function so that once a player's king is checkmated, the player's other pieces remain the same positions and the player will be skipped in each turn.